Minority-Focused Text-to-Image Generation via Prompt Optimization

Supplementary Material

6. Theoretical Results

6.1. Proof of Proposition 1

Proposition 1. *The objective function in Eq. (7) is equivalent (upto a constant factor) to the negative ELBO w.r.t.* $\log p_{\boldsymbol{\theta}}(\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) \mid \mathcal{C})$ *when integrated over timesteps with* $\bar{w}_s \coloneqq \alpha_s/(1-\alpha_s)$*:*

$$\sum_{s=1}^{T} \bar{w}_s \mathcal{J}_{\mathcal{C}}(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) = \sum_{s=1}^{T} \mathbb{E}_{\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{z}_{s|t,0}, \mathcal{C})\|_2^2] \quad (10)$$
$$\gtrapprox -\log p_{\boldsymbol{\theta}}(\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) \mid \mathcal{C}),$$

where $\boldsymbol{z}_{s|t,0} \coloneqq \sqrt{\alpha_s}\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) + \sqrt{1-\alpha_s}\boldsymbol{\epsilon}$.

Proof. Remember the definition of the objective function in Eq. (7):

$$\mathcal{J}_{\mathcal{C}}(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) \coloneqq \mathbb{E}_{\boldsymbol{\epsilon}}\big[\|\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) - \hat{\boldsymbol{z}}_0(\boldsymbol{z}_{s|t,0}, \mathcal{C})\|_2^2\big].$$

Plugging this into the LHS of Eq. (10) yields:

$$\begin{aligned}
&\sum_{s=1}^{T} \bar{w}_s \mathcal{J}_{\mathcal{C}}(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) \\
&= \sum_{s=1}^{T} \frac{\alpha_s}{1-\alpha_s} \mathbb{E}_{\boldsymbol{\epsilon}}\big[\|\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) - \hat{\boldsymbol{z}}_0(\boldsymbol{z}_{s|t,0}, \mathcal{C})\|_2^2\big] \\
&= \sum_{s=1}^{T} \frac{\alpha_s}{1-\alpha_s} \mathbb{E}_{\boldsymbol{\epsilon}}\Bigg[\Bigg\| \frac{1}{\sqrt{\alpha_s}}(\boldsymbol{z}_{s|t,0} - \sqrt{1-\alpha_s}\boldsymbol{\epsilon}) \\
&\qquad - \frac{1}{\sqrt{\alpha_s}}(\boldsymbol{z}_{s|t,0} - \sqrt{1-\alpha_s}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{z}_{s|t,0}, \mathcal{C}))\Bigg\|_2^2\Bigg] \\
&= \sum_{s=1}^{T} \frac{\alpha_s}{1-\alpha_s} \mathbb{E}_{\boldsymbol{\epsilon}}\left[\left\| \frac{\sqrt{1-\alpha_s}}{\sqrt{\alpha_s}}(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{z}_{s|t,0}, \mathcal{C}))\right\|_2^2\right] \\
&= \sum_{s=1}^{T} \mathbb{E}_{\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{z}_{s|t,0}, \mathcal{C})\|_2^2],
\end{aligned} \quad (11)$$

where the second equality is from the definitions of $\boldsymbol{z}_{s|t,0}$ and $\hat{\boldsymbol{z}}_0(\boldsymbol{z}_{s|t,0}, \mathcal{C})$:

$$\boldsymbol{z}_{s|t,0} \coloneqq \sqrt{\alpha_s}\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) + \sqrt{1-\alpha_s}\boldsymbol{\epsilon}$$
$$\hat{\boldsymbol{z}}_0(\boldsymbol{z}_{s|t,0}, \mathcal{C}) \coloneqq \frac{1}{\sqrt{\alpha_s}}(\boldsymbol{z}_{s|t,0} - \sqrt{1-\alpha_s}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{z}_{s|t,0}, \mathcal{C})).$$

Note that the last expression in Eq. (11), which is the same as the RHS of Eq. (10), is equivalent (up to a constant) to the expression of the negative ELBO w.r.t. $\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}})$ [20, 28]. The distinction here is that now we use a text-conditional diffusion model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\cdot, \mathcal{C})$ that approximates $\log p_{\boldsymbol{\theta}}(\cdot|\mathcal{C})$. This completes the proof. □

6.2. Theoretical issues on Eq. (6)

We continue from Sec. 3.3 to scrutinize the theoretical challenges that arise in the naively-extended optimization framework in Eq. (6). To proceed, we first restate the objective function in Eq. (6):

$$\mathcal{J}(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) \coloneqq \mathbb{E}_{\boldsymbol{\epsilon}}\big[\|\hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) - \mathtt{sg}(\hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_{s|t,0}^w, \mathcal{C}_{\boldsymbol{v}}))\|_2^2\big].$$

Remember that we identified three theoretical issues that impair the connection to the target log-likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{z}_0|\mathcal{C})$: (i) the reliance on the CFG-based clean predictions; (ii) obstructed gradient flow through the second term in the squared-L2 loss; and (iii) the incorporation of $\mathcal{C}_{\boldsymbol{v}}$ within the second term in the loss.

CFG-based clean prediction. We start by examining the first point, the pathology due to the CFG-based clean predictions. Suppose we incorporate the CFG-based clean predictions $\hat{\boldsymbol{z}}_0^w$ in our framework Eq. (7), in place of the non-CFG terms $\hat{\boldsymbol{z}}_0$. The objective function then becomes:

$$\mathcal{J}_{\mathcal{C}}^w(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) \coloneqq \mathbb{E}_{\boldsymbol{\epsilon}}\big[\|\hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) - \hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_{s|t,0}^w, \mathcal{C})\|_2^2\big].$$

To see its connection to log-likelihood, let us consider the weighted sum of this objective with $\bar{w}_s \coloneqq \alpha_s/(1-\alpha_s)$ (as in Proposition 1). Manipulating the averaged objective similarly as in Sec. 6.1 then yields:

$$\begin{aligned}
&\sum_{s=1}^{T} \bar{w}_s \mathcal{J}_{\mathcal{C}}^w(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) \\
&= \sum_{s=1}^{T} \bar{w}_s \mathbb{E}_{\boldsymbol{\epsilon}}\big[\|\hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) - \hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_{s|t,0}^w, \mathcal{C})\|_2^2\big] \\
&= \sum_{s=1}^{T} \frac{\alpha_s}{1-\alpha_s} \mathbb{E}_{\boldsymbol{\epsilon}}\Bigg[\Bigg\| \frac{1}{\sqrt{\alpha_s}}(\boldsymbol{z}_{s|t,0}^w - \sqrt{1-\alpha_s}\boldsymbol{\epsilon}) \\
&\qquad - \frac{1}{\sqrt{\alpha_s}}(\boldsymbol{z}_{s|t,0}^w - \sqrt{1-\alpha_s}\tilde{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}^w(\boldsymbol{z}_{s|t,0}^w, \mathcal{C}))\Bigg\|_2^2\Bigg] \\
&= \sum_{s=1}^{T} \mathbb{E}_{\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \tilde{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}^w(\sqrt{\alpha_s}\hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) + \sqrt{1-\alpha_s}\boldsymbol{\epsilon}, \mathcal{C})\|_2^2].
\end{aligned} \quad (12)$$

Observe that in the RHS of Eq. (12), we see the CFG noise estimation term $\tilde{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}^w$, instead of $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ as in Eq. (11). This comes from the use of $\hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_{s|t,0}^w, \mathcal{C})$ in the second term of the squared-L2 loss. Since $\tilde{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}^w$ represents a distinct probability density, say $\tilde{p}_{\boldsymbol{\theta}}(\cdot \mid \mathcal{C})$, the averaged objective in Eq. (12) is no longer connected to our focused conditional log-likelihood $\log p_{\boldsymbol{\theta}}(\cdot \mid \mathcal{C})$.

One may wonder whether the use of CFG for the first term in the squared-L2 loss of Eq. (7) is safe. However, we claim that it is also problematic. To show this, we derive the associated log-likelihood, which is immediate with the algebra used for Eq. (12):

$$\sum_{s=1}^{T} \bar{w}_s \mathbb{E}_{\boldsymbol{\epsilon}}\big[\|\hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) - \hat{\boldsymbol{z}}_0(\boldsymbol{z}^w_{s|t,0}, \mathcal{C})\|_2^2\big] \gtrapprox -\log p_{\boldsymbol{\theta}}(\hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) \mid \mathcal{C}).$$

We see that now the diffusion model (represented by $p_{\boldsymbol{\theta}}$) should estimate the conditional log-density w.r.t. the CFG clean prediction $\hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}})$. We argue that this estimation may be inaccurate, since the CFG clean sample in the T2I context is potentially off-manifold. As analyzed in Chung et al. [7], the CFG clean prediction $\hat{\boldsymbol{z}}_0^w(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}})$ is in fact an extrapolation between $\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}})$ and $\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t)$ (controlled by w). As a result, it may deviate from the data manifold, particularly for high w values commonly used in standard T2I scenarios; see Figure 3 in Chung et al. [7] for details. This off-manifold issue is especially pronounced during the initial phase of inference, as also reported in other studies [27]. See Tab. 2 for experimental results that support this claim.

Obstructed gradient. Now we move onto the second issue. From the above analysis, we saw that the noise prediction in the second term is crucial for relating the objective function to the log-likelihood, meaning that allowing gradient flow through the second term is essential for accurate likelihood optimization. However, blocking the gradient via the stop-gradient on the second term contradicts this theoretical intuition. We found that the use of stop-gradient actually degrades performance; see Tab. 2 for instance.

$\mathcal{C}_{\boldsymbol{v}}$ in the second term. The reasoning behind the third challenge follows naturally from the previous analyses. In this case, the corresponding log-likelihood term can be derived as:

$$\sum_{s=1}^{T} \bar{w}_s \mathbb{E}_{\boldsymbol{\epsilon}}\big[\|\hat{\boldsymbol{z}}(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) - \hat{\boldsymbol{z}}_0(\boldsymbol{z}_{s|t,0}, \mathcal{C}_{\boldsymbol{v}})\|_2^2\big] \gtrapprox -\log p_{\boldsymbol{\theta}}(\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t, \mathcal{C}_{\boldsymbol{v}}) \mid \mathcal{C}_{\boldsymbol{v}}).$$

We see that $\mathcal{C}_{\boldsymbol{v}}$ appears in conditioning variable, which diverges from our interest of approximating $\log p_{\boldsymbol{\theta}}(\cdot \mid \mathcal{C})$. See Tab. 2 for experimental results that corroborate this.

7. Supplementary Details

7.1. Details on the metric in Um and Ye [52]

We continue from Sec. 3.3 to provide additional details on the likelihood metric developed by Um and Ye [52]. This original version is defined on pixel space $\boldsymbol{x}_0 \in \mathbb{R}^d$ (rather than latent domain $\boldsymbol{z}_0 \in \mathbb{R}^k$ as ours), formally written as [52]:

$$\mathcal{J}(\boldsymbol{x}_t; s) \coloneqq \mathbb{E}_{\boldsymbol{\epsilon}}\big[d(\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t), \hat{\boldsymbol{x}}_0(\boldsymbol{x}_{s|t,0}))\big],$$

where $\boldsymbol{x}_t$ is a noisy pixel-domain image, and $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t)$ represents a clean estimate of $\boldsymbol{x}_t$: $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t) \coloneqq (\boldsymbol{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}'_{\boldsymbol{\theta}}(\boldsymbol{x}_t))/\sqrt{\alpha_t}$, where $\boldsymbol{\epsilon}'_{\boldsymbol{\theta}}$ denotes a pixel diffusion model (different from our $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$). Here $\boldsymbol{x}_{s|t,0}$ indicates a noised version of $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t)$ according to timestep s: $\boldsymbol{x}_{s|t,0} \coloneqq \sqrt{\alpha_s}\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t) + \sqrt{1-\alpha_s}\boldsymbol{\epsilon}$, and $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{s|t,0})$ is a denoised version of $\boldsymbol{x}_{s|t,0}$. d indicates a discrepancy metric (*e.g.*, LPIPS [58]). This quantity is interpretable as a reconstruction loss of $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t)$, and theoretically, it is an estimator of the negative log-likelihood of $\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t)$ [52].

Similar to ours, the authors in Um and Ye [52] employs this metric as a guidance function for minority sampling, sharing similar spirit as ours. In doing so, they propose several techniques such as stop-gradient, learning-rate scheduling, and the incorporation of LPIPS as d. Their proposed metric for the guidance function is expressible as:

$$\mathcal{J}(\boldsymbol{x}_t; s) \coloneqq \eta_t \mathbb{E}_{\boldsymbol{\epsilon}}\big[\texttt{LPIPS}(\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t), \texttt{sg}(\hat{\boldsymbol{x}}_0(\boldsymbol{x}_{s|t,0})))\big], \tag{13}$$

where η_t indicates learning rate at time t designed to decrease over time, and `LPIPS` is the perceptual metric proposed by Zhang et al. [58]. Although this approach offers considerable advantages in traditional image generation tasks (such as unconditional generation), it is not optimized for T2I generation, which presents unique challenges and requires more specialized techniques. This is confirmed by our experimental results, where a straightforward extension of their framework yields only modest performance improvements. See Tab. 2 and Tab. 4b for details.

7.2. Implementation details

Pretrained models and baselines. We employed the official checkpoints provided in HuggingFace for all three pretrained models. For the null-prompted DDIM baselines, we employed "commonly-looking" as the null-text prompt for all three pretrained models. The CADS baselines were primarily obtained using the recommended settings in the paper [42], while we adjusted the hyperparameters on SDXL-Lightning for adaptation to distilled models. Specifically, we set $\tau_1 = 0.8$, $\tau_2 = 1.0$, and $s = 0.1$, while keeping other settings unchanged. For SGMS, we respected the original design choices (like the use of `sg`) and tuned the remaining hyperparameters to attain the optimal performance in the T2I context. In particular, we used the squared-L2 loss as the discrepancy metric and employed $s = 0.75T$. For their latent optimizations, we employed Adam optimizer [24] (as ours) with learning rates between 0.005 and 0.01. Similar to ours, latent updates were performed intermittently, with $N = 3$ (*i.e.*, one update per three sampling steps). Each

Target	CS ↑	IR ↑	LL ↓
Text	31.3503	0.2406	0.9263
Null-text	31.1089	0.1575	1.0175
Token (ours)	**31.6465**	**0.2744**	**0.9006**

(a) Influence of optimization target

Method	CS ↑	IR ↑	LL ↓
Unoptimized	31.4395	0.1845	1.0465
Naive (Eq. (6))	30.2994	-0.1944	0.9245
Ours (Eq. (7))	**31.7369**	**0.2839**	**0.9230**

(b) Impact of objective function $\mathcal{J}$

Type	CS ↑	IR ↑	LL ↓
Default	31.5154	0.2492	0.9355
Gaussian	31.5054	0.2405	0.9429
Word init	**31.7369**	**0.2839**	**0.9230**

(c) Effect of initializing $\boldsymbol{v}$

Table 4. **Impact of key design choices.** "CS" denotes ClipScore [17], and "IR" is Image-Reward [53]. 'LL' indicates log-likelihood. "Text" is the optimization framework focused on updating the text-embedding $\mathcal{C}$, and "Null-text" refers to the one that adjusts the null-text embedding (as in [33]). "Unoptimized" corresponds to the standard DDIM sampler. "Default" denotes the case that simply employs the default embedding assigned with an added learnable token, while "Gaussian" initializes $\boldsymbol{v}$ from a multivariate Gaussian distribution constructed using the mean and variance of the token embeddings from the text-encoder $\mathcal{T}$. "Word init" indicates initializing with a specific word embedding. We used SDv1.5 for the results herein.

Target	CS ↑	IR ↑	LL ↓
Eq. (7)	31.3658	0.7207	0.5449
Eq. (8)	**31.4194**	**0.7331**	0.5449

(a) Influence of `sg`-trick

Type	CS ↑	IR ↑	LL ↓
$s = 0.75T$	31.4534	0.1569	0.9469
$s = T - t$	**31.7369**	**0.2839**	**0.9230**

(b) Impact of adaptive s

Type	CS ↑	IR ↑	LL ↓
$\mathcal{C}$	31.7548	0.2293	0.9744
$\mathcal{C}_{\boldsymbol{v}^*}$	**31.7871**	**0.2858**	**0.9511**

(c) Effect of using $\mathcal{C}$

Table 5. **Effectiveness of our new techniques.** "$\mathcal{C}$" refers to the use of $\mathcal{C}$ during sampling steps without prompt optimization (when incorporating an intermittent prompt update, *i.e.*, $N > 1$). On the other hand, "$\mathcal{C}_{\boldsymbol{v}^*}$" refers to the use of optimized token embeddings in the latest steps. Our results show that the proposed design choices consistently outperform naive approaches. The results in (a) were obtained using SDXL-Lightning, while SDv1.5 was employed for (b) and (c).

latent optimization consisted of three distinct update steps: $K = 3$.

Evaluations. The ClipScore values reported in our paper were due to `torchmetrics`[3]. For PickScore and Image-Reward, we employed the implementations provided in the official code repositories[4][5]. Precision and Recall were computed with $k = 5$ using the official codebase of Han et al. [16][6]. The computations of Density and Coverage were based on the authors' official codebase[7]. The log-likelihood values were evaluated based on the implementation of Hong et al. [21][8]. In-Batch Similarity that we used in the diversity optimization (in Tab. 3) were computed with the repository of Corso et al. [8][9].

Hyperparameters. Our results were obtained using $s = T - t$, and we used Adam optimizer with $K = 3$, similar to SGMS. Learning rates were set between 0.001 and 0.002 across all experiments. We shared the same intermittent update rate of $N = 3$ with SGMS. For initializing $\boldsymbol{v}$, we shared the same word embedding for "cool" for the main results (presented in Tab. 1). The number of learnable tokens for our approaches was set to 1. As described in Sec. 3.3, we globally used $\lambda = 1$ across all experiments. For the experiments on SDXL-Lightning that involves two distinct text-encoders, we employed a single Adam optimizer to jointly update both embedding spaces to minimize parameter complexity. We also synchronized other design choices for the two encoders, *e.g.*, sharing the same initial token embedding.

Computational complexity. The inference time for DDIM is approximately 1.136 seconds per sample, with CADS requiring a similar amount of time. The complexities of SGMS and our approach are rather higher due to the inclusion of backpropagation and iterative updates of latents or prompts. Specifically, SGMS takes 5.756 seconds per sample, while our sampler requires slightly more time – 6.205 seconds per sample – which we attribute to the additional backpropagation pass introduced by our removal of gradient-blocking. All computations herein were performed on SDv2.0 using a single NVIDIA A100 GPU.

Other details. Our implementation is based on PyTorch [37], and experiments were performed on twin NVIDIA A100 GPUs. Code is available at https://github.com/soobin-um/MinorityPrompt.

8. Ablations, Analyses, and Discussions

8.1. Additional ablation studies

Tab. 4 investigates the impact of some key design choices in our framework. Specifically, Tab. 4a highlights the benefits of optimizing small sets of token embeddings, which outperform alternatives targeting text or null-text embeddings in both text alignment and log-likelihood. The advantage of using the proposed objective function Eq. (7) is exhibited

[3] https://lightning.ai/docs/torchmetrics/stable/multimodal/clip_score.html
[4] https://github.com/yuvalkirstain/PickScore
[5] https://github.com/THUDM/ImageReward
[6] https://github.com/hichoe95/Rarity-Score
[7] https://github.com/clovaai/generative-evaluation-prdc
[8] https://github.com/unified-metric/unified_metric
[9] https://github.com/gcorso/particle-guidance

Init word	CS ↑	IR ↑	LL ↓
"uncommon"	31.6971	0.2825	**0.8868**
"special"	31.6178	**0.2922**	0.9342
"cool"	**31.7369**	0.2839	0.9230

(a) Sensitivity to the initial word

Position	CS ↑	IR ↑	LL ↓
–	31.4395	0.1845	1.0465
Prefix	31.5519	0.2809	0.9249
Postfix	**31.7369**	**0.2839**	**0.9230**

(b) Impact of the position of $\mathcal{S}$

# of tokens	CS ↑	IR ↑	LL ↓
1	**31.6465**	**0.2744**	**0.9006**
2	31.5866	0.2204	0.9163
4	31.4989	0.2679	0.9419

(c) Effect of # of learnable tokens

Table 6. **Exploring the design space of learnable tokens.** "Init word" indicates the word embedding used for initializing $\boldsymbol{v}$. "–" refers to standard DDIM sampling without prompt optimization. "Prefix" denotes prepending the placeholder string $\mathcal{S}$ to $\mathcal{P}$, while "Postfix" indicates appending it to the end of $\mathcal{P}$. "# of tokens" represents the number of tokens assigned to the string $\mathcal{S}$. We observe that the proposed approach is not highly sensitive to the choice of initial word, and as suggested, attaching $\mathcal{S}$ at the end of the prompts yields the best performance. Additionally, using a single token is sufficient to achieve performance gains. We used SDv1.5 for the results herein.

Method	CLIPScore ↑	PickScore ↑	ImageReward ↑	Precision ↑	Recall ↑	Density ↑	Coverage ↑	Likelihood ↓
DDIM-SDE (*i.e.*, Eq. (14))	**31.5806**	**21.5693**	0.2451	**0.5840**	0.6940	**0.6772**	**0.8040**	1.1666
+ MinorityPrompt	31.5002	21.3508	**0.2907**	0.5070	**0.7030**	0.5452	0.7820	**1.0069**
DPM-Solver++(2M) [32]	31.4447	**21.4659**	0.2161	**0.5930**	0.7160	**0.6666**	**0.8520**	0.9744
+ MinorityPrompt	**31.8595**	21.3827	**0.3035**	0.5510	**0.7446**	0.5482	0.7910	**0.8522**
CFG++ [7]	31.4755	**21.4490**	0.1938	**0.5710**	0.7100	**0.6400**	**0.8470**	1.0452
+ MinorityPrompt	**31.7627**	21.3399	**0.3062**	0.5540	**0.7284**	0.5394	0.7670	**0.9183**

Table 7. **Compatibility with existing ODE/SDE solvers.** The term "DDIM-SDE" indicates a stochastic DDIM sampler [46] (see Eq. (14) for definition), where we used 50 sampling steps with a CFG weight of $w = 7.5$. "DPM-Solver++(2M)" is a fast ODE solver introduced by Lu et al. [32]; for this, we adhered to the recommended settings of 25 steps with a CFG weight of $w = 7.5$. "CFG++" represents a DDIM sampler featuring enhanced CFG mixing recently proposed by Chung et al. [7], for which we incorporated the authors' suggested parameters: 50 steps with a guidance weight of $\lambda = 0.6$. We emphasize that MinorityPrompt exhibits robust compatibility and substantial performance improvements when integrated with existing solvers, encompassing both ODE and SDE frameworks. All results were derived using SDv1.5.

in Tab. 4b, where the naively-extended framework based on Eq. (6) demonstrates significant performance gap compared to our carefully-crafted approach. Tab. 4c explores various initialization techniques for $\boldsymbol{v}$. While all methods yield substantial improvements over the unoptimized sampler (see "unoptimized" in Tab. 4b for comparison), we observe that further gains can be achieved with properly chosen initial words.

Tab. 5 explores the impact of our techniques developed for further improvements in Sec. 3.3. We see consistent enhancements over naive design choices. A key insight from Tab. 5c is that reusing token embeddings optimized at earlier timesteps, denoted as "$\mathcal{C}_{\boldsymbol{v}^*}$" in the table, offers limited benefit compared to simply using the base prompts $\mathcal{C}$. This finding highlights the evolutionary nature of our prompt-tuning framework, which supports continual updates to embeddings across sampling timesteps.

Tab. 6 investigates the design choices related to learnable tokens in our framework. Observe in Tab. 6a that our framework consistently delivers significant performance gains across different initial word embeddings. Regarding the position of $\mathcal{S}$, appending it to the end of the prompts yields better results. We speculate that prepending may have a greater impact on the semantics of the text embeddings due to the front-weighted nature of the training process for the

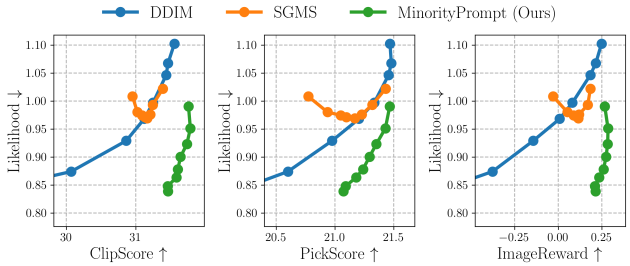

Figure 5. **Trade-off analysis.** The DDIM curves were calculated using a range of CFG weights. In particular, we employed: $w \in \{1.0, 2.0, \ldots, 5.0, 7.5, 9.0, 12.5\}$. For the SGMS baseline [52], we fixed the CFG weight as $w = 7.5$ and swept the learning rate (*i.e.*, η_t in Eq. (13)) over $[2 \times 10^{-3}, 2 \times 10^{-2}]$. Similarly for MinorityPrompt, we shared the same CFG weight of $w = 7.5$ while controlling the learning rate (used with `AdamGrad` in Algorithm 2) over $[5 \times 10^{-4}, 4 \times 10^{-3}]$. We highlight that our trade-off is significantly more favorable compared to the baselines that suffer from substantial degradation when attempting to generate low-likelihood samples. We employed SDv1.5 for obtaining the curves.

CLIP text encoders [39] employed in our T2I models. As exhibited in Tab. 6c, a single token is sufficient to realize the performance benefits of our approach. The performance

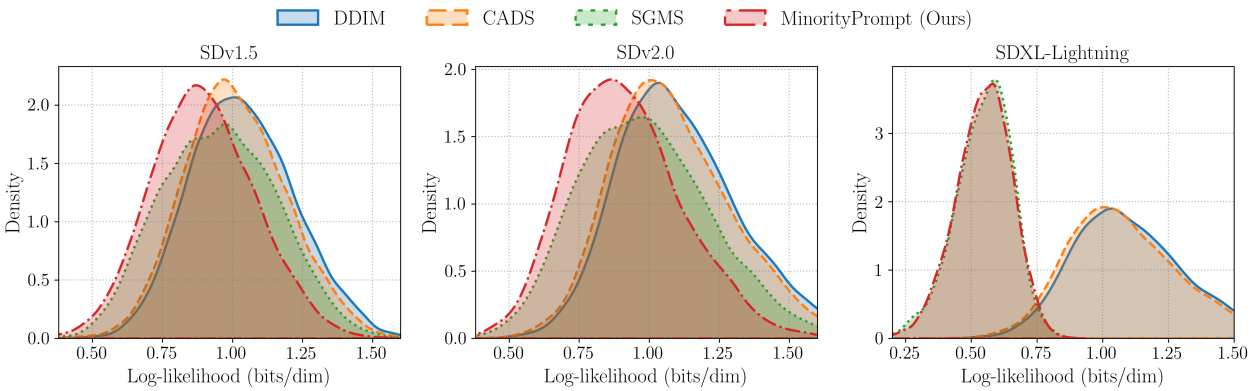

Figure 6. **Comparison of log-likelihood distributions.** The likelihood values were measured using the PF-ODE-based computation proposed by Song et al. [48]. We observe that MinorityPrompt better produces low-likelihood instances compared to the considered baselines across all three pretrained models.

degradation observed with increasing tokens is likely due to their heightened influence on semantics, similar to the effect of $\mathcal{S}$'s position.

Tab. 7 evaluates the performance of MinorityPrompt when integrated with existing ODE/SDE solvers. Specifically, we investigate three notable solvers: (i) DDIM-SDE, (ii) DPM-Solver++(2M)[32], and (iii) CFG++[7]. DDIM-SDE denotes a stochastic version of the DDIM sampler, formally written as [46]:

$$\boldsymbol{z}_{t-1} = \sqrt{\alpha_{t-1}}\hat{\boldsymbol{z}}_0(\boldsymbol{z}_t) + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{z}_t) + \sigma_t\boldsymbol{\epsilon}_t, \tag{14}$$

where $\sigma_t := \sqrt{(1-\alpha_{t-1})/(1-\alpha_t)}\sqrt{1-\alpha_t/\alpha_{t-1}}$ and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We omit the dependence of $\hat{\boldsymbol{z}}_0$ and $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ on $\mathcal{C}$ for simplicity.

Observe in Tab. 7 that MinorityPrompt consistently enhances the baseline versions of each solver, exhibiting the same trend we saw in our main results (*e.g.*, in Tab. 1) evaluated with the baseline DDIM. This demonstrates the robustness and adaptability of our approach, further emphasizing its practical significance as a method that can be seamlessly integrated with a range of powerful solvers. The results also demonstrate that our framework is effective even with a small number of sampling steps (*e.g.*, 25 steps), as evidenced by its superior performance when applied to DPM-Solver++(2M).

8.2. Trade-off analysis

We explore the trade-off characteristics of our framework controlled by the learning rate (used with `AdamGrad` in Algorithm 2). For comparison, we present the trade-off performances of two key baselines: DDIM and SGMS [52]. For DDIM, we evaluate the performance across various CFG weights w to examine its ability to produce low-likelihood instances. While for the SGMS baseline, we adjusted the learning rate (*i.e.*, η_t in Eq. (13)) with a fixed CFG weight of $w = 7.5$. The evaluation of ours was conducted under a similar condition as SGMS, with a fixed CFG weight (*i.e.*, $w = 7.5$) and varying learning rates.

Method	Text alignment ↑	Uniqueness ↑	Image quality ↑
DDIM	4.1159	2.6029	**3.8725**
DDIM + null	3.8986	2.6841	3.7768
CADS [42]	3.8029	2.9362	3.6696
SGMS [52]	3.6667	3.1217	2.9246
MInorityPrompt	**4.2145**	**4.2812**	3.8565

Table 8. **User study results.** We present human evaluation results focusing on three key aspects: (i) Text alignment, (ii) Uniqueness (i.e., the degree of minority representation), and (iii) Image quality. Feedback was collected from 23 participants, where they were asked to rate 15 image sets each containing generated outputs by 5 distinct methods. Ratings were provided on a scale from 1 to 5. We see a consistent performance benefit of ours, with a significant improvement in the uniqueness of generated samples while maintaining text alignment and image quality.

Fig. 5 shows the trade-off performances of the considered three approaches. Observe that the trade-off achieved by our framework significantly outperforms the baselines that experience substantial quality degradation when generating low-likelihood instances. A notable point is that SGMS often enters a degeneration regime at high learning rates, where further increases fail to yield lower-likelihood samples. In contrast, our framework does not exhibit such pathological behavior, demonstrating the robustness of the proposed approach compared to existing baselines. Also, we see the effect of controlling the learning rate within our framework: a higher learning rate tends to produce instances with lower likelihoods, accompanied by some degradation in text alignment and sample quality.

8.3. Limitations and discussion

A disadvantage is that our framework introduces additional computational costs (similar to [52]), particularly when compared to standard samplers like DDIM. As noted in Sec. 7.2, this is mainly due to the incorporation of backpropagation and iterative updates of prompts. Additionally, the removal of gradient-blocking, aimed at restoring the theoretical connection to the target conditional density, further contributes to the overhead. Future work could focus on optimizing these processes to reduce computational demands. One potential approach is to develop an approximation of our objective that mitigates the need for extensive backpropagation while maintaining its alignment with the target log-likelihood.

9. Additional Experimental Results

9.1. Log-likelihood distributions

Fig. 6 exhibits the log-likelihood distributions for MinorityPrompt and the baseline models across all three pretrained architectures. We see that MinorityPrompt consistently produces lower log-likelihood instances, further demonstrating its improved capability of generating minority samples. The distributions for SDXL-Lightning are more dispersed than in other scenarios, which may be attributed to the larger latent space upon which SDXL-Lightning is based. The competitive results compared to SGMS observed in SDXL-Lightning may arise from the limited optimization opportunities available in distilled models (as discussed in the manuscript).

9.2. User preference study

For a more comprehensive evaluation of our approach from the perspective of human preference, we conducted a user study based on participant feedback. Specifically, we asked 23 participants to evaluate 15 sets of images generated by five distinct methods, rating each image set on three key aspects: (i) Text alignment, (ii) Uniqueness (i.e., the degree of minority representation), and (iii) Image quality. Ratings were provided on a scale from 1 to 5.

Tab. 8 exhibits the detailed user study results. Notably, MinorityPrompt demonstrates a consistent performance advantage, as observed in the main results (e.g., Tab. 1), by significantly enhancing the uniqueness of generated samples with only marginal compromises in text alignment and image quality. This further validates the effectiveness of our approach as a text-to-image minority generator.

9.3. Additional generated samples

To facilitate a more comprehensive qualitative comparison among the samplers, we provide an extensive showcase of generated samples for all the focused T2I pretrained models. See Figures 7-9 for details. In addition, we exhibit samples generated using various diversity-focused approaches (discussed at the end of Sec. 4.2); see Fig. 10 for further details.

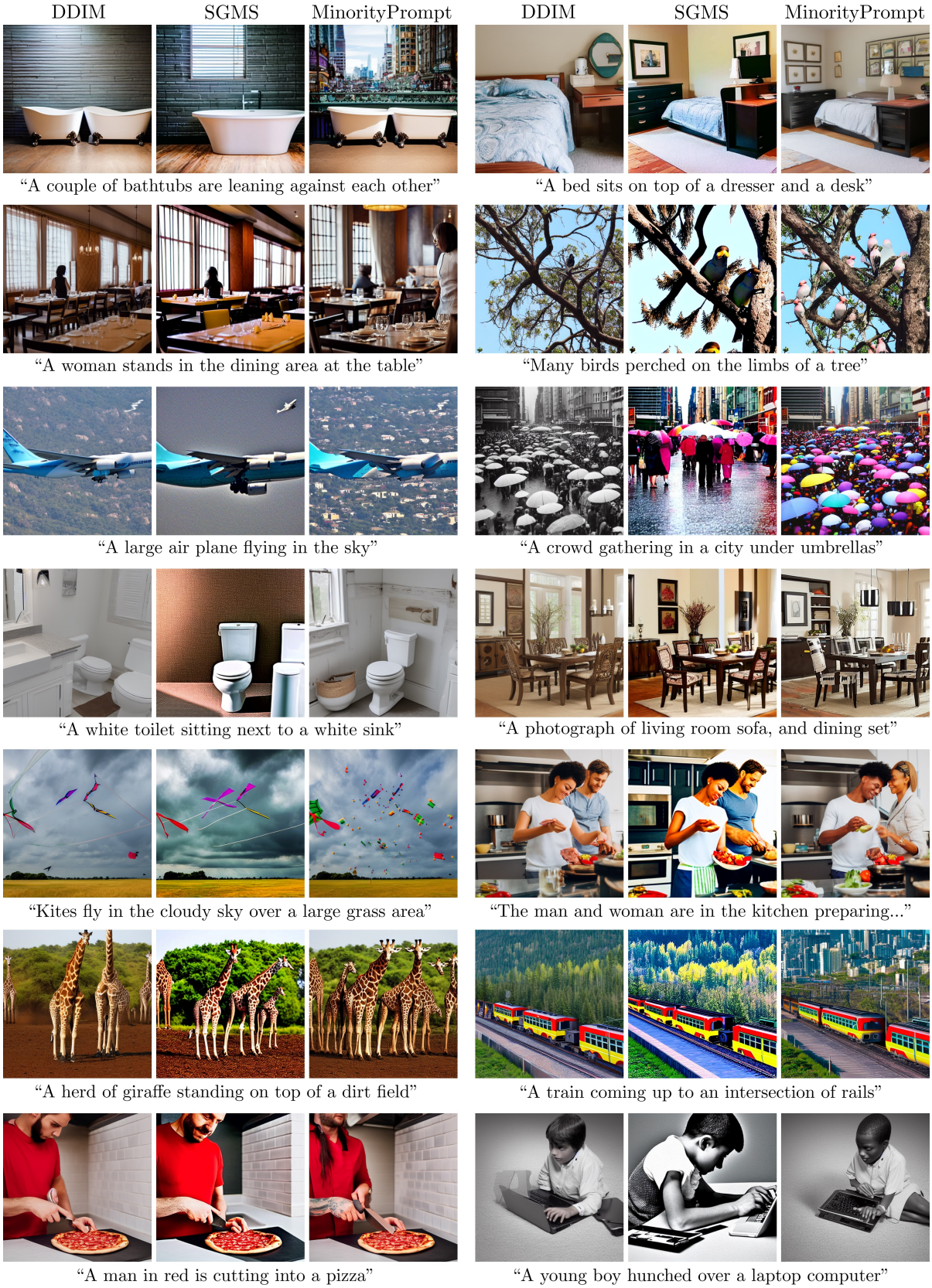

Figure 7. **Generated samples on SDv1.5.** Generated samples from three distinct samplers: (i) DDIM [46]; (ii) SGMS [52]; (iii) MinorityPrompt (ours). Random seeds were shared across all three methods.

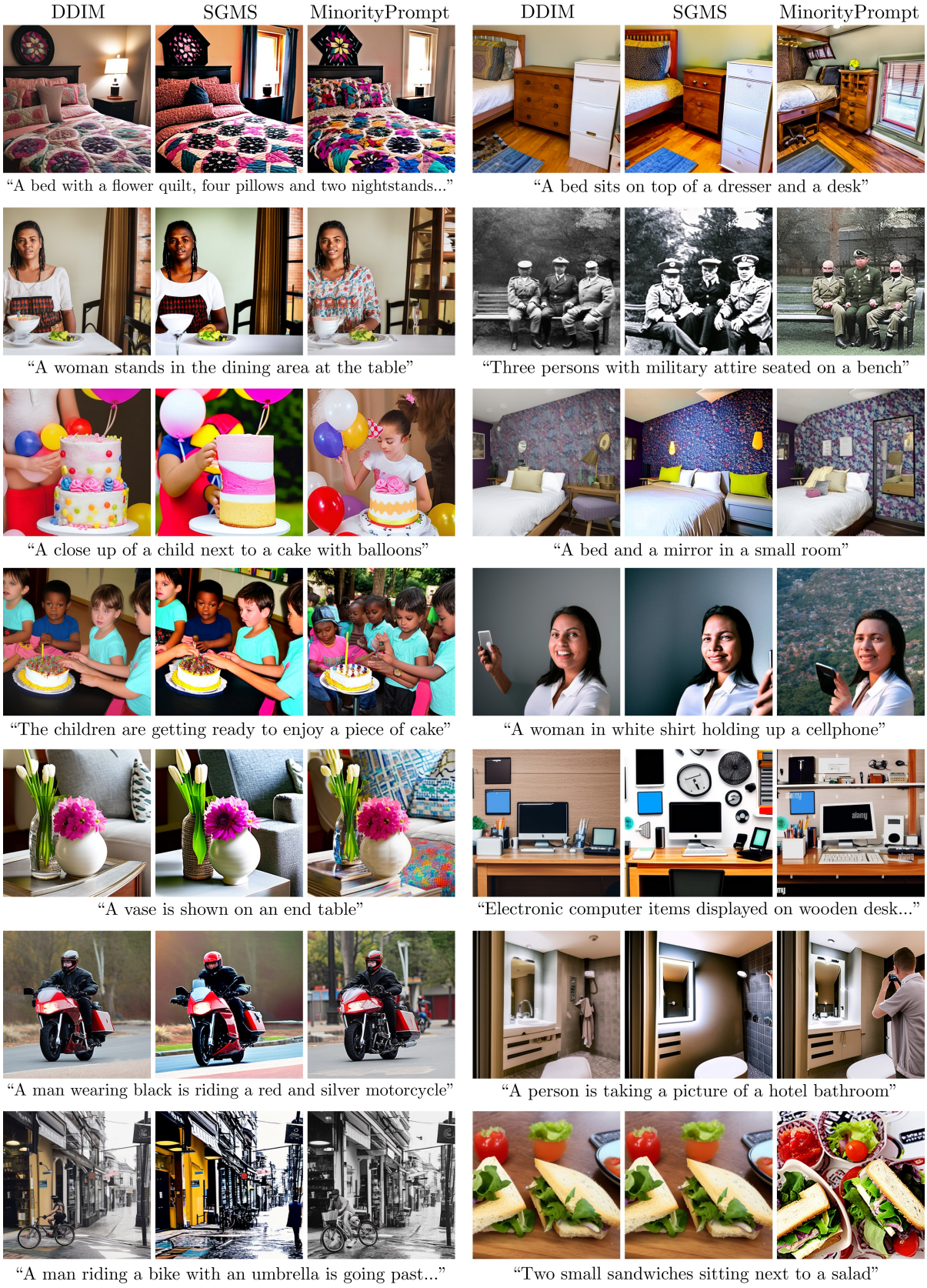

Figure 8. **Generated samples on SDv2.0.** Generated instances from three different techniques: (i) DDIM [46]; (ii) SGMS [52]; (iii) MinorityPrompt (ours). We shared the same random seeds across all three approaches.

Figure 9. **Additional generated samples on SDXL-Lightning.** Generated samples from three different approaches: (i) DDIM [46]; (ii) SGMS [52]; (iii) MinorityPrompt (ours). We employed the same initial noises across all three samplers.

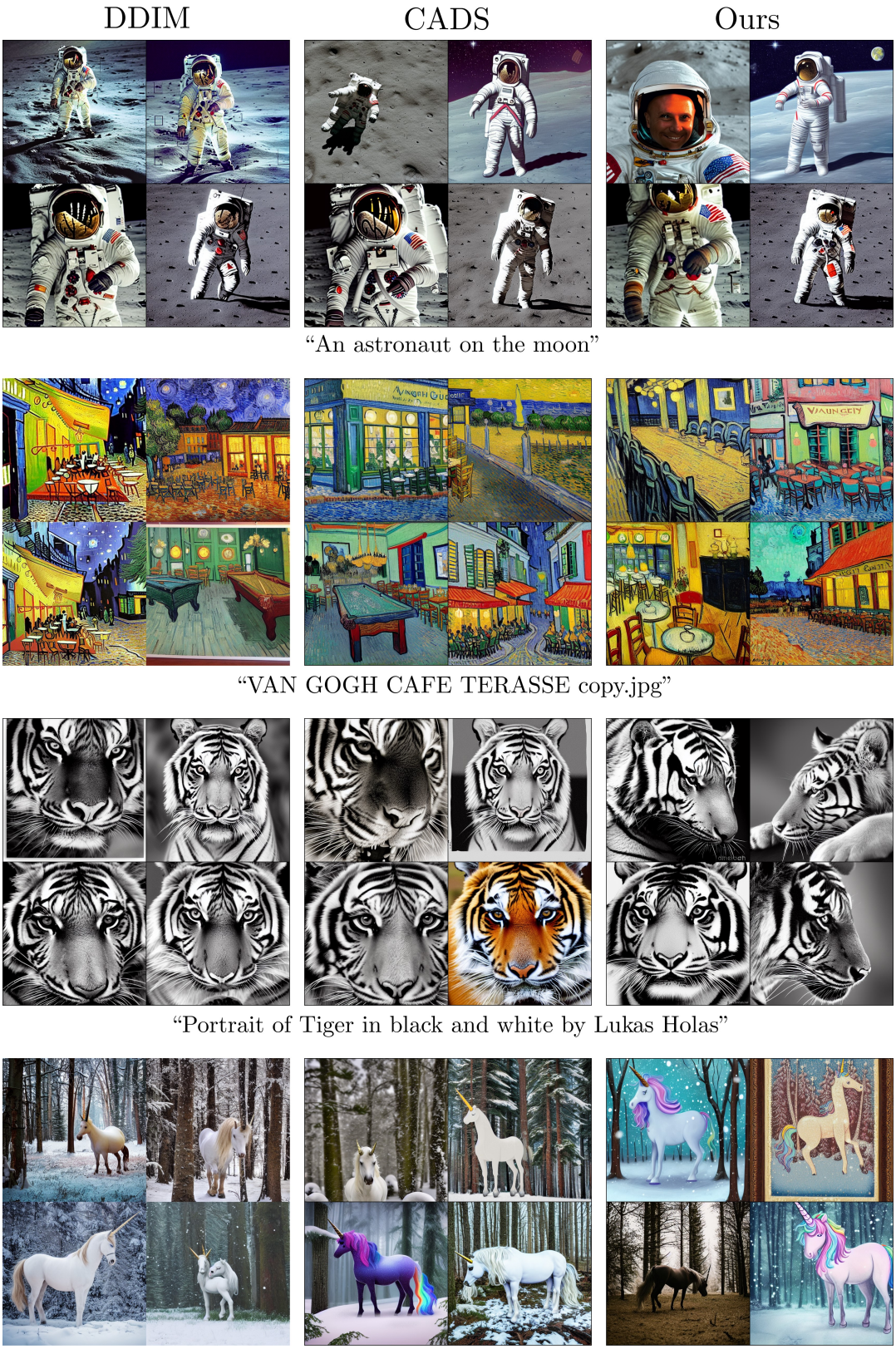

Figure 10. **Generated samples from diversity-enhancing approaches on SDv1.5.** Samples generated using three distinct methods: (i) DDIM [46]; (ii) CADS [42]; (iii) Ours (*i.e.*, Eq. (9)). All samplers shared the same initial noise for generation.